RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

LEHRSTUHL INFORMATIK XI SOFTWARE FÜR EINGEBETTETE SYSTEME

**Professor Dr.-Ing. Stefan Kowalewski**
Dipl.-Inform. Bastian Schlich

Aachen, 20. Dezember 2004
SWS: V2/Ü2

# Systemprogrammierung

WS 2004/2005

Übungsblatt 10

Abgabe der Lösungen: ab 10. Januar 2005 in den Übungen

---

### Aufgabe 1 (3+2=5 Punkte): Scheduling Strategien

Gegeben sei folgendes Scheduling mit vier Prioritätsklassen. Jeder Klasse ist eine eigene Warteschlange und ein eigenes Schedulingverfahren zugeordnet:

| Klasse | Schedulingverfahren | Priorität |
|---|---|---|
| 0 | RR mit Quantum 1 Zeiteinheit | höchste |
| 1 | RR mit Quantum 4 Zeiteinheiten | |
| 2 | RR mit Quantum 16 Zeiteinheiten | |
| 3 | FCFS | niedrigste |

(RR = Round Robin; FCFS = First Come First Served)

Es wird *Process Aging* verwendet, d.h. wenn das Zeitquantum eines Prozesses abgelaufen ist, wird er an das Ende der Warteschlange mit nächstniedriger Priorität gestellt. Neuen Prozessen ist eine bestimmte Priorität zugeordnet. Sie werden an das Ende der Warteschlange ihrer Prioritätsklasse angehängt. Es wird am Ende jeder Zeiteinheit überprüft welcher Prozess am Anfang der nicht leeren Warteschlange mit der höchsten Priorität steht und dieser dann im nächsten Schritt bearbeitet.

Folgende Prozesse sollen betrachtet werden:

| Prozess | Ankunftszeit | Priorität | Laufzeit |
|---|---|---|---|
| A | 0 | 0 | 4 |
| B | 0 | 1 | 7 |
| C | 1 | 0 | 1 |
| D | 2 | 1 | 4 |
| E | 5 | 3 | 20 |
| F | 10 | 1 | 3 |
| G | 13 | 0 | 2 |
| H | 15 | 1 | 3 |
| I | 16 | 1 | 3 |

Ein Prozess, der zu einem Zeitpunkt $t$ ankommt, wird in der $(t+1)$-ten Zeiteinheit berücksichtigt.

a) Geben Sie für die ersten 20 Zeiteinheiten an, welchem Prozess Rechenzeit zugeteilt wird.

b) Geben Sie für alle Prioritätsklassen an, welche Prozesse sich nach Ablauf der ersten 20 Zeiteinheiten noch in der Warteschlange befinden (in der korrekten Reihenfolge), wie viele Zeiteinheiten jeder einzelne Prozess noch laufen muss und wie viele Zeiteinheiten von seinem Quantum noch übrig sind.

**Aufgabe 2 (2+2+2+1=7 Punkte): Speicherverwaltung**

a) Nehmen Sie einen Rechner an, der mit 128 MByte Hauptspeicher ausgerüstet ist. Dieser Speicher ist im Moment leer und wird komplett mit einem Buddy-System verwaltet. Sinnvollerweise wird die Speicherbelegung mit Hilfe eines Binärbaumes dargestellt. Dabei stellt die Wurzel den gesamten Speicher von 128 MByte dar, die nächste Ebene zwei Segmente mit jeweils 64 MByte usw. Bei einer Speicheranforderung beginnt der Suchalgorithmus in der untersten passenden Ebene und sucht freie Segmente bereits „angebrochener" Buddies. Ist kein solcher halbbenutzter Teilbaum auffindbar, geht der Algorithmus in die nächsthöhere Ebene und sucht dort einen angebrochenen Teilbaum. Bei Erfolg wird dieser Buddy bis auf die benötigte Größe „hinuntergeteilt", das entstandene Segment als belegt gekennzeichnet und die nächste Anforderung bearbeitet. Stellt der Algorithmus fest, dass sämtliche möglichen Segmente auf der derzeitigen Ebene belegt oder unterteilt sind, bricht er die Suche erfolglos ab. Bei Freigabe von Speicherplätzen werden die Buddies bis zur höchstmöglichen Ebene rekombiniert.

   Gehen wir jetzt von einem völlig leeren Speicher aus. Der Reihe nach werden folgende Partitionen angefordert oder freigegeben:

| *Operation* | *Partition* | *Größe* |
|---|---|---|
| Anforderung | A | 6 MB |
| Anforderung | B | 4 MB |
| Anforderung | C | 8 MB |
| Anforderung | D | 16 MB |
| Anforderung | E | 32 MB |
| Freigabe | A | |
| Anforderung | F | 16 MB |
| Freigabe | E | |
| Freigabe | B | |

   Stellen Sie die einzelnen Schritte als Binärbaum dar (Beginn nur mit Wurzel)! Hat der Algorithmus die Wahl zwischen zwei Wegen, so wähle er stets den linken.

b) Nun werden gewichtete Buddies betrachtet. Hierbei wird ein Knoten nicht in zwei gleich große Äste aufgeteilt, sondern Knoten, deren Größe ohne Rest durch 3 teilbar ist, werden in Buddies im Verhältnis 1:2 aufgeteilt. Sind die Knoten durch 4, aber nicht durch 3 teilbar, werden Buddies im Größenverhältnis 1:3 erzeugt.

   Führen Sie auch hier die obigen Operationen aus, beginnend mit einem leeren Speicher von 128 MByte. Bei Speicheranforderung teilt der Algorithmus den kleinsten noch passenden Buddy. Stellen Sie die einzelnen Schritte wieder als Binärbaum dar.

c) Gegeben sei die folgende Ausgangssituation:

| *Operation* | *Partition* | *Größe* |
|---|---|---|
| Anforderung | E | 4 MB |
| Anforderung | F | 8 MB |
| Anforderung | G | 4 MB |
| Anforderung | H | 8 MB |
| Anforderung | I | 16 MB |
| Anforderung | J | 8 MB |
| Freigabe | A | |
| Freigabe | C | |
| Freigabe | E | |

In dieser Darstellung wählt der Algorithmus jeweils den oberen Zweig, falls zwei Teilbäume äquivalent sind (also entspricht hier der obere Zweig dem linken Zweig in einer Darstellung, bei der die Wurzel oben steht).Sämtliche Anforderungen kommen zunächst in eine Warteschlange. Betrachtet werden folgende zwei Verfahren:

1) FIFO: Die Warteschlange wird stur von vorne nach hinten abgearbeitet (unabhängig von der Größe der Anforderungen).
2) Best Fit: Alle Anforderungen in der Warteschlange besitzen gleiche Priorität. Ist Speicherplatz frei, so wird er gemäß Best Fit mit der Anforderung aus der Warteschlange belegt, die am besten paßt. Kleine Anforderungen sollen dabei gegenüber großen Anforderungen bevorzugt werden. Anschließend sollen die Freigaben erfolgen.

Geben Sie für beide Verfahren den jeweiligen Endzustand des Binärbaums an sowie eventuell noch in der Warteschlange befindliche Anforderungen!

d) Überlegen Sie sich, wie man den Buddy-Algorithmus verbessern könnte, um eine bessere Speichernutzung zu erreichen!

**Aufgabe 3 (1+1+2=4 Punkte): Speicherverwaltung**

Ein großer Vorteil der virtuellen Speicherverwaltung ist, daß der Programmierer sich nicht um den physikalischen Adreßraum kümmern muß, sondern mit logischen Adressen arbeiten kann. Spätestens bei der Ausführung eines Programmes müssen diese logischen Adressen aber in physikalische Adressen umgewandelt werden.

Das Umsetzungsprinzip beim *Segmenting* funktioniert wie folgt: Jedem Programm ist eine Segmenttabelle zugeordnet, in der für jedes logische Segment die physikalische Anfangsadresse `b` und die Größe des Segments `l` eingetragen ist, die das Programmsegment im Speicher einnimmt.

Eine logische Adresse besteht nun aus der Segmentnummer `s`, mit der aus der Segmenttabelle die entsprechende Anfangsadresse `b` bestimmt wird, und einem Offset `d`, der angibt, an welcher Stelle in Relation zur Anfangsadresse sich die Adresse, auf die man zugreifen will, befindet. Dabei wird geprüft, ob der Offset kleiner als die Segmentlänge `l` ist, da anderenfalls eine ungültige Adresse angesprochen wird. Die physikalische Adresse ergibt sich also zu `b+d`.

Für die Speicherverwaltung nach dem Segmentierungsverfahren ist für ein Programm folgende Segmenttabelle gegeben (Länge in Speicherworten):

| Segment | Basis | Länge |
|---|---|---|
| 0 | 1410 | 365 |
| 1 | 400 | 70 |
| 2 | 500 | 120 |
| 3 | 630 | 515 |
| 4 | 1145 | 150 |

a) Wieviele Speicherworte stehen dem Programm im physikalischen Speicher zur Verfügung?

b) Welches ist die kleinste und welches die größte für das Programm verfügbare physikalische Adresse?

c) Berechnen Sie zu den folgenden physikalischen Adressen jeweils die logischen Adressen:

1. 762
2. 1145
3. 1146
4. 485

**Aufgabe 4 (1+1+1+1+1=5 Punkte): Speicherverwaltung**

Ein Programm besteht aus fünf Seiten im virtuellen Adressraum und hat drei Seiten im Hauptspeicher zur Verfügung. Es werden die Speicherseiten in folgender Reihenfolge referenziert:

$$\omega = 15245345242143$$

Geben Sie die Belegung des Hauptspeichers nach jedem Speicherzugriff an und verwenden Sie folgende Paging-Algorithmen:

1. First-In-First-Out
2. Least-Recently-Used
3. Clibm
4. Second Chance
5. Optimale Verdrängung

Zählen Sie jeweils die Anzahl der Seitenfehler und vergleichen Sie die Ergebnisse.

**Aufgabe 5 (1+1+1+1=4 Punkte): Speicherverwaltung**

Die Hauptspeicherverwaltung eines Betriebssystems teilt bei Speicheranforderung Speicherbereiche variabler Länge zu. Die Verwaltung dieser Freispeicherbereiche geschieht durch eine Liste (s. Abb. 1). In dieser Freispeicherliste befinden sich aktuell vier Bereiche. Nacheinander treffen folgende fünf Anforderungen ein:

**25 40 30 50 45**

Folgende Belegungsstrategien sollen für die Zuteilung angewandt werden:

a) First-Fit

b) Best-Fit

c) Worst-Fit

d) Next-Fit
   Next-Fit funktioniert wie First-Fit, nur merkt sich der Algorithmus die letzte genutzte Position und beginnt beim nächsten Mal an dieser Stelle.

Geben Sie für jede Strategie die nach jeder erfolgten Zuweisung resultierende Freispeicherliste in einer Tabelle an und zeichnen Sie die Belegung der in der Liste enthaltenen Speicherelemente nach der letzten Zuweisung in die Liste ein.

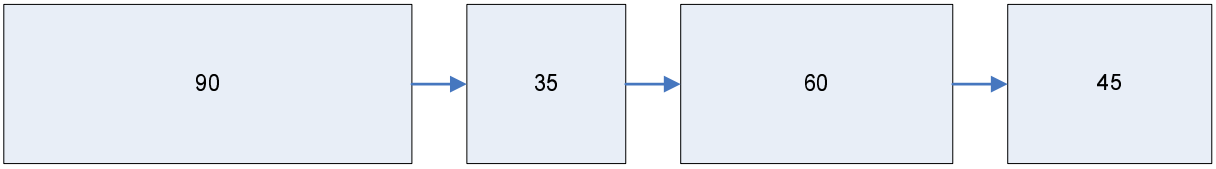


Abbildung 1: Die Freispeicherliste des Bestriebssystems